Modular Structural Operational Semantics (Modular SOS, or MSOS)

- Peter Mosses (1999)
- Addresses the non-modularity aspects of SOS
  - A definitional framework *is non-modular* when, in order to add a new feature to an existing language, one needs to revisit and change some of the already defined, unrelated language features
  - The non-modularity of SOS will become clear when we define IMP++
- Why modularity is important
  - Modifying existing rules when new rules are added is *error prone*
  - When *experimenting* with language design, one needs to make changes quickly; having to do unrelated changes slows us down
  - *Rapid language development*, e.g., domain-specific languages

Philosophy of MSOS

- *Separate the syntax* from configurations and treat it differently
- Transitions go from *syntax to syntax*, hiding the other configuration components into *transition labels*
- Labels encode all the non-syntactic configuration changes
- Specialized notation in transition labels, to
  - Say that certain configuration components stay unchanged
  - Say that certain configuration changes are propagated from the premise to the conclusion of a rule

MSOS Transitions

- An MSOS transition has the form

$$P \xrightarrow{\Delta} P'$$

  - *P* and *P'* are programs or tragments of program
  - Δ is a label describing the changes in the configuration components, defined as a record; primed fields stay for "after" the transition
- Example:

$$x := i \xrightarrow{\{\text{state}=\sigma,\ \text{state}'=\sigma[i/x],\ \ldots\}} \texttt{skip}$$

  - This rule can be automatically "desugared" into the SOS rule

$$\langle x := i, \sigma \rangle \to \langle \texttt{skip}, \sigma[i/x] \rangle$$

But also into (if the configuration contains more components, like in IMP++)

$$\langle x := i, \sigma, \omega \rangle \to \langle \texttt{skip}, \sigma[i/x], \omega \rangle$$

MSOS Labels

- Labels are field assignments, or records, and can use "…" for "and so on", called *record comprehension*
- Fields can be primed or not.
  - Unprimed = configuration component *before* the transition is applied
  - Primed = configuration component *after* the transition is applied
- Some fields appear both unprimed and primed (called read-write), while others appear only primed (called write-only) or only unprimed (called read-only)

MSOS Labels

- Field types
  - Read/write = fields which appear both unprimed and unprimed

$$x := i \xrightarrow{\{\text{state}=\sigma,\ \text{state}'=\sigma[i/x],\ \ldots\}} \texttt{skip}$$

  - Write-only = fields which appear only primed

$$\texttt{output}\ (i) \xrightarrow{\{\text{output}'=i,\ \ldots\}} \texttt{skip}$$

  - Read –only = fields which appear only unprimed (see rules next)

$$\frac{e_2 \xrightarrow{\text{env}=\rho[v_1/x]} e_2'}{\text{let } x = v_1 \text{ in } e_2 \xrightarrow{\text{env}=\rho} \text{let } x = v_1 \text{ in } e_2'}$$

MSOS Rules

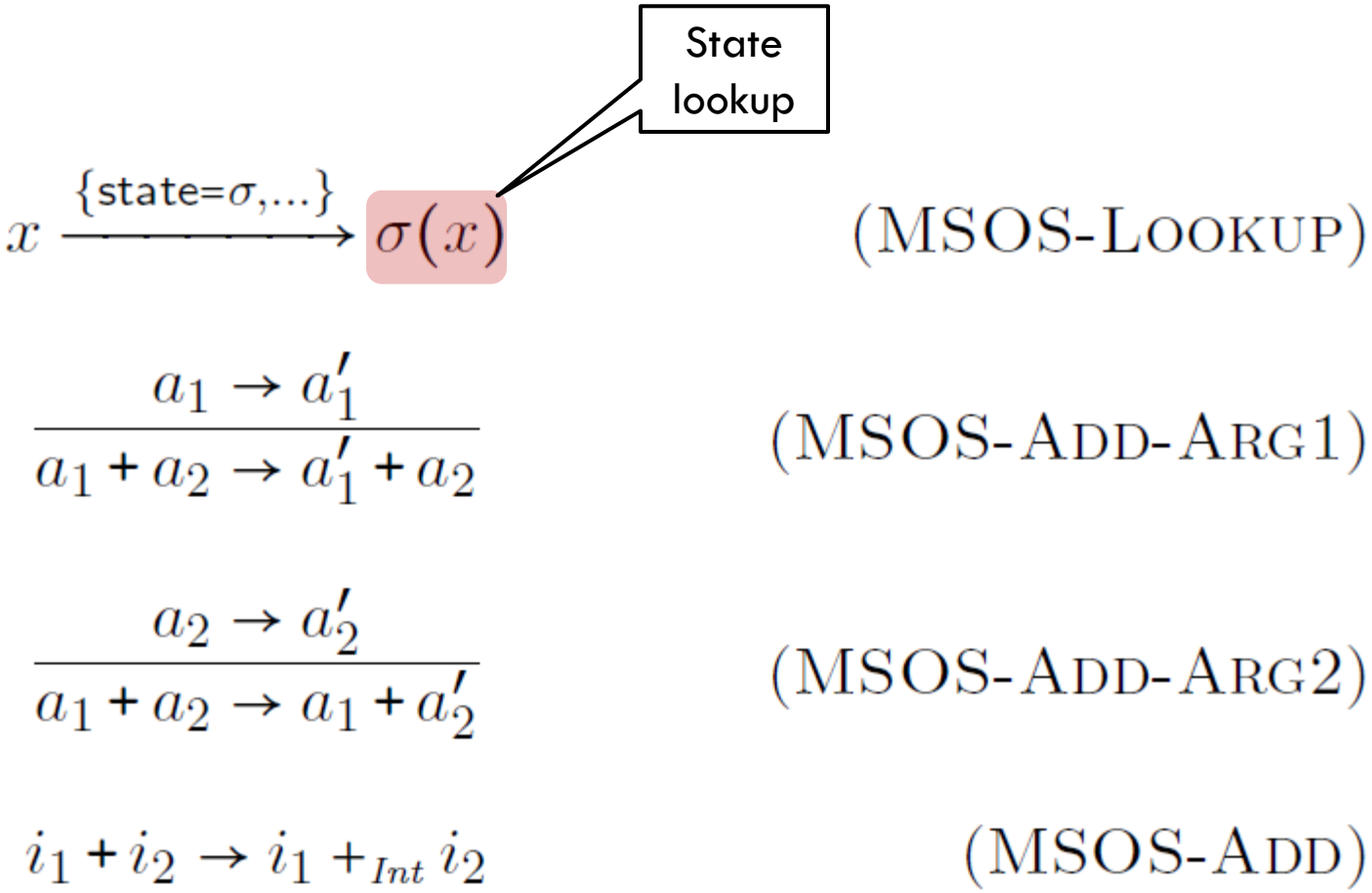
- Like in SOS, but using MSOS transitions as sequents
- Same labels or parts of them can be used multiple times in a rule
- Example:

$$\frac{s_1 \xrightarrow{\Delta} s_1'}{s_1 \; ; \; s_2 \xrightarrow{\Delta} s_1' \; ; \; s_2}$$

  - Same Δ means that changes propagate from premise to conclusion
- The author of MSOS now promotes a simplifying notation
  - If the premise and the conclusion repeat the same label or part of it, simply drop that label or part of it. For example:

$$\frac{s_1 \rightarrow s_1'}{s_1 \; ; \; s_2 \rightarrow s_1' \; ; \; s_2}$$

MSOS of IMP - Arithmetic



MSOS of IMP - Arithmetic

$$\frac{a_1 \to a_1'}{a_1 \,/\, a_2 \to a_1' \,/\, a_2} \qquad \text{(MSOS-Div-Arg1)}$$

$$\frac{a_2 \to a_2'}{a_1 \,/\, a_2 \to a_1 \,/\, a_2'} \qquad \text{(MSOS-Div-Arg2)}$$

$$i_1 \,/\, i_2 \to i_1 \,/_{Int}\, i_2 \quad \text{when } i_2 \neq 0 \qquad \text{(MSOS-Div)}$$

MSOS of IMP - Boolean

$$\frac{a_1 \to a_1'}{a_1 <= a_2 \to a_1' <= a_2} \qquad \text{(MSOS-LEQ-ARG1)}$$

$$\frac{a_2 \to a_2'}{i_1 <= a_2 \to i_1 <= a_2'} \qquad \text{(MSOS-LEQ-ARG2)}$$

$$i_1 <= i_2 \to i_1 \leq_{Int} i_2 \qquad \text{(MSOS-LEQ)}$$

MSOS of IMP - Boolean

$$\frac{b \to b'}{\texttt{not}\, b \to \texttt{not}\, b'} \quad \text{(MSOS-Not-Arg)}$$

$$\texttt{not true} \to \texttt{false} \quad \text{(MSOS-Not-True)}$$

$$\texttt{not false} \to \texttt{true} \quad \text{(MSOS-Not-False)}$$

$$\frac{b_1 \to b_1'}{b_1\, \texttt{and}\, b_2 \to b_1'\, \texttt{and}\, b_2} \quad \text{(MSOS-And-Arg1)}$$

$$\texttt{false and}\, b_2 \to \texttt{false} \quad \text{(MSOS-And-False)}$$

$$\texttt{true and}\, b_2 \to b_2 \quad \text{(MSOS-And-True)}$$

MSOS of IMP - Statements

$$\frac{a \to a'}{x := a \to x := a'} \quad \text{(MSOS-Asgn-Arg2)}$$

$$x := i \xrightarrow{\{\text{state}=\sigma,\ \text{state'}=\sigma[i/x],\ \ldots\}} \texttt{skip} \quad \text{(MSOS-Asgn)}$$

$$\frac{s_1 \to s_1'}{s_1 \texttt{ ; } s_2 \to s_1' \texttt{ ; } s_2} \quad \text{(MSOS-Seq-Arg1)}$$

$$\texttt{skip ; } s_2 \to s_2 \quad \text{(MSOS-Seq-Skip)}$$

MSOS of IMP - Statements

$$\frac{b \to b'}{\texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2 \to \texttt{if}\ b'\ \texttt{then}\ s_1\ \texttt{else}\ s_2} \quad \text{(MSOS-If-Arg1)}$$

$$\texttt{if true then}\ s_1\ \texttt{else}\ s_2 \to s_1 \quad \text{(MSOS-If-True)}$$

$$\texttt{if false then}\ s_1\ \texttt{else}\ s_2 \to s_2 \quad \text{(MSOS-If-False)}$$

$$\texttt{while}\ b\ \texttt{do}\ s \to \texttt{if}\ b\ \texttt{then}\ (s\ \texttt{;}\ \texttt{while}\ b\ \texttt{do}\ s)\ \texttt{else skip} \quad \text{(MSOS-While)}$$

$$\texttt{vars}\ xl\ \texttt{;}\ s \xrightarrow{\{\text{state'}= xl \mapsto 0, \ldots\}} s \quad \text{(MSOS-Vars)}$$

MSOS in Rewriting Logic

- Any MSOS can be associated a rewrite logic theory (or, equivalently, a Maude module)
- Idea:
  - Desugar MSOS into SOS
  - Apply the SOS-to-rewriting-logic representation, but
  - Hold the non-syntactic configuration components in an ACI-data-structure, so that we can use ACI matching to retrieve only the fields of interest (which need to be read or written to)

MSOS of IMP in Maude

- See file **`imp-semantics-msos.maude`**